Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

SYSDE

SYSDE Computación

"Apoyo para la Sistematización del Proceso de Desarrollo de Software"

y

"Apoyo para la Implementación del Marco de Gestión de Activos del Software Factory de SYSDE"

Informe Final de Proyecto de Práctica para optar por el grado de Bachiller en Ingeniería en Computación

Jorge Eduardo Jaime Castro

Barreal de Heredia, 2006

RESUMEN

En el presente documento se describe la estrategia para satisfacer los alcances planteados en los proyectos "Apoyo para la Sistematización del Proceso de Desarrollo de Software" y "Apoyo para la Implementación del Marco de Gestión de Activos del Software Factory de SYSDE". Los alcances de estos proyectos variaron en gran parte durante el transcurso de la practica, por lo que los alcances aquí mencionados, no son los que se encuentran planteados en las sinopsis de los proyectos.

Debido al cambio en los alcances y la falta de tiempo, algunos no se pudieron realizar y, otros por motivos de confidencialidad no se describen en este informe, pues se refieren a documentos propios de los procesos de la empresa.

En las secciones uno y dos se describen los pasos a seguir para desarrollar el proyecto "Apoyo para la Implementación del Marco de Gestión de Activos del Software Factory de SYSDE", que al final quedo con solo tres alcances basados en documentación e investigación de los procesos de respaldo del repositorio. Dentro de los pasos descritos en el primer alcance se encuentra la especificación y análisis de la herramienta de automatización de respaldos, el diseño de la misma continua en la sección tres.

En las secciones cuatro y cinco se describen los pasos para desarrollar el proyecto de "Apoyo para la Sistematización del Proceso de Desarrollo de Software", en estas secciones se describen los antecedentes objetivos y alcances del proyecto. Este proyecto al igual que el anterior sufrió modificaciones en sus alcances quedando apenas con tres grandes alcances. El alcance dos no se logro satisfacer por falta de tiempo, y el alcance tres no se especifica pues este solo consistió en crear algunas pantallas del Sistema SYSDE Fábrica. En la sección donde se especifica el primer alcance de este proyecto se describe la especificación y análisis de la herramienta para el mantenimiento de los archivos de recurso. Su Diseño se encuentre especificado en la sección siete.

En las sección seis se describen los pasos que se utilizaron para proveer la propiedad multilenguaje a el sistema SYSDE Fabrica. Se describen los dos métodos planteados y la escogencia del mismo.

TABLA DE CONTENIDOS

1 GESTIÓN DE ACTIVOS

1.1 Naturaleza del Proyecto

Uno de los componentes más importantes para SYSDE como empresa desarrolladora de software, son sus activos del software factory. Estos activos adquieren relevancia en la organización pues constituyen la base para la generación de nuevos productos, nuevas versiones y por lo tanto nuevos negocios para SYSDE.

Para la gestión de estos activos, se ha creado un área especializada con el objetivo primordial de conservar de forma integral, organizada y protectora de todos los activos del software factory, para lo cual se ha definido un Marco de Gestión de Activos, que describe los procedimientos y pautas para la administración de los activos.

Para cumplir con los procedimientos y las pautas definidas en el Marco de Gestión de Activos se deberá investigar y desarrollar una serie de herramientas(políticas, manuales, software) para la administración efectiva de estos activos tan valiosos para SYSDE. Por lo tanto, se ha creado un proyecto cuyo objetivo consistirá en crear las herramientas de software requeridas para facilitar la gestión de los activos de software.

1.2 Antecedentes

SYSDE como empresa desarrolladora de soluciones de negocio y tecnología para entidades financieras, microfinancieras y operadoras de pensión, se ha destacado como líder en estos mercados.

Parte de ese éxito ha sido la innovación en aplicar las mejores prácticas de la industria del software para la implementación de las soluciones alrededor del mundo. Actualmente SYSDE, ha implementado un software factory, que para tales efectos ha requerido de elementos importantes como infraestructura, metodología, herramientas y una organización que le

permita cumplir con sus objetivos. Una de las unidades creadas para apoyar el software factory, es la Unidad de Gestión de Activos.

La Unidad de Gestión de Activos de Software Factory, tiene el objetivo de conservar de forma integral, organizada y protectora de todos los activos del software factory.

Los activos del software factory, según el *Software Engineering Institute* (SEI), constituyen las bases para la producción de productos en una línea de producción de software. Partiendo de esta definición, se consideran activos del software factory el conjunto de los componentes de ensambles para generar una versión, típicamente serán los módulos, componentes de programación y herramientas que facilitan la construcción de nuevos módulos y su respectiva documentación, manual e instalador que forma parte de una versión.

Todos estos activos serán normados por el Marco de Gestión de Activos del Software Factory, el cual determina las pautas y procedimientos de cómo debe aplicarse la metodología de YSDE
para la gestión, desarrollo e implementación de los sus soluciones tecnológicas.

Definido y publicado el Marco de Gestión de Activos de Software Factory, se deberán investigar y desarrollar herramientas para la gestión eficiente de los activos de software factory.

1.3 Objetivos

Apoyar en la implementación de tareas específicas del Marco de Gestión de Activos, tales como: automatización de tareas de respaldos, comprobación de respaldos, entre otros. Desarrollar herramientas para una administración eficiente de los artefactos de la metodología.

Generación de instaladores de versiones de los productos de SYSDE, utilizando el Install Shield.

Documentación formal de las políticas de gestión de activos. Todas estas actividades estarán contribuyendo con la Unidad de Gestión de Activos de Software Factory para la implementación del Marco de Gestión de Activos.

1.4 Alcances

Documentar y automatizar los procesos de administración del repositorio de activos. Este proceso involucra:

- Documentar el procedimiento de respaldos
- Definir esquemas de seguridad para accesos internos y externos al repositorio.
- Definir y crear estructuras en el repositorio para todos los productos
- Definición de esquemas de contingencia y recuperación del repositorio.
- Administrar y documentar políticas de crecimiento y depuración del repositorio.
- Crear una herramienta para realizar los respaldos automáticos

Generar los distintos instaladores de los productos SAF y el instalador beta SYSDE Pensión y SYSDE Banca

Actualizar el Marco de Gestión de Activos SYSDE SAF, acorde a los procedimientos documentados en la administración del repositorio y mediante un análisis GAP entre el Marco de Gestión de Activos SYSDE SAF y sistemas basados en Developer generar el borrador del Marco de Gestión de Activos para los sistemas Developer.

2 ESPECIFICACIÓN DEL PROYECTO SEGÚN ALCANCES DE GESTION DE ACTIVOS

2.1 ALCANCE #1

En esta sección se describen los pasos para satisfacer el siguiente alcance: Documentar y automatizar los procesos de administración del repositorio de activos. Este proceso involucra:

2.1.1 Proceso de documentación

Todos los documentos se generan a base de la estructura que posee SYSDE para los mismos.

- Documentar el procedimiento de respaldos
    - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.
    - Investigar sobre el uso y manejo de la herramienta utilizada para mantener el repositorio de activos, el Visual SourceSafe. Generando un documento son aspectos útiles para los respaldos utilizando el manual del Visual SourceSafe
    - Investigar el proceso actual de respaldos, documentando los pasos significativos.
    - Listar los posibles pasos, para un eventual esquema de respaldos. Comentarlos con el líder de proyecto.
    - A base de los pasos listados y revisados y la información recopilada, generar el documento sobre el esquema de respaldos.

- Definir esquemas de seguridad para accesos internos y externos al repositorio.

    - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.

    - Investigar sobre el uso y manejo de la herramienta utilizada para mantener el repositorio de activos, el Visual SourceSafe. Generando un documento son aspectos útiles para los respaldos utilizando el manual del Visual SourceSafe

    - Investigar el esquema actual de seguridad, documentando los pasos significativos.

    - Listar los posibles pasos, para un eventual esquema de seguridad. Comentarlos con el líder de proyecto.

    - A base de los pasos listados y revisados y la información recopilada, generar el documento sobre el esquema de seguridad.

- Definir y crear estructuras en el repositorio para todos los productos

    - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.

    - Investigar sobre el uso y manejo de la herramienta utilizada para mantener el repositorio de activos, el Visual SourceSafe. Generando un documento son aspectos útiles para los respaldos utilizando el manual del Visual SourceSafe

- Listar los diferentes tipos de instrumentos (documentación) utilizados para la fabricación de los productos de SYSDE. Además de la estructura de las fuentes y ejecutables que generan las herramientas de programación que utiliza SYSDE para crear sus productos.

- Estudiar el esquema actual, planteado en el marco de gestión de activos.

- Establecer las diferencias entre los Tipos de instrumentos fuentes y ejecutables de los diferentes productos de SYSDE.

- Mediante las diferencias, y la información recopilada proponer una estructura para el repositorio de activos.

- Definición de esquemas de contingencia y recuperación del repositorio.

  - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.

  - Investigar sobre el uso y manejo de la herramienta utilizada para mantener el repositorio de activos, el Visual SourceSafe. Generando un documento con aspectos útiles para los respaldos utilizando el manual del Visual SourceSafe

  - Investigar el proceso actual de recuperación del repositorio, documentando los pasos significativos.

  - Listar los posibles pasos, para un eventual esquema de recuperación del repositorio. Comentarlos con el líder de proyecto.

- A base de los pasos listados y revisados y la información recopilada, generar el documento sobre el esquema de recuperación del repositorio.

- Administrar y documentar políticas de crecimiento y depuración del repositorio.

  - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.

  - Investigar sobre el uso y manejo de la herramienta utilizada para mantener el repositorio de activos, el Visual SourceSafe. Generando un documento con aspectos útiles para los respaldos utilizando el manual del Visual SourceSafe

  - Investigar las políticas de crecimiento y depuración del repositorio actual, documentar los aspectos más significativos.

  - Investigar sobre el manejo que hace la herramienta sobre las versiones de los archivos contenidos en el repositorio, documentando los aspectos que tengan que ver con la depuración.

  - A base de los del conocimiento adquirido crear un documento con las políticas de depuración.

- Crear una herramienta para realizar los respaldos automáticos

2.1.1 Herramienta de automatización Visual SourceSafe

OBJETIVO GENERAL

Realizar un programa que realice los respaldos automáticos del repositorio de activos que es administrado por el Visual SourceSafe, utilizando las herramientas de consola que brinda este.

OBJETIVOS ESPECIFICOS

- Administrar la lista de las bases de datos que se desean respaldar.

- Listar los proyectos de cada base de datos para generar los respaldos de cada base de datos y generar el respaldo con esta lista,.

- Realizar los respaldos guardándolos en carpetas que en su nombre referencia la fecha en que se realizo el respaldó.

DESCRIPCIÓN DEL PROYECTO

Realizar un programa que realice los respaldos automáticos de las bases de datos del repositorio de activos utilizando la herramienta de consola que provee el Visual SourceSafe. Este programa deberá mantener una lista de las bases de datos que se desean respaldar.

Con base en la lista abrir las bases de datos, listar sus proyectos, en caso de que la base de datos exceda los 5Gb deberán agruparse esos proyectos. Estos respaldos

deberán guardarse en una carpeta con un nombre que contenga la fecha del respaldo, esta carpeta deberá estar en la ruta especificada por el usuario.

Antes de generar los respaldos el sistema deberá verificar si existe espacio en la unidad, donde se guardaran los respaldo de lo contrario tendrá que buscarlo en las otras unidades.

ALCANCES

1. Mantener una lista que el usuario pueda modificar de todas las bases de datos que se desean respaldar

2. Abrir las bases de datos, listar los proyectos de estas, con base a esto generar el respaldo de las mismas.

3. Crear una carpeta donde se guarden los respaldos referenciando en su nombre la fecha del respaldo, verificar el espacio en disco, buscar particiones con espacio, para guardar los respaldos.

4. Generar una bitácora de incidencias, en la realización de los respaldos.

5. Generar un archivo por lote que pueda realizar los respaldos de las bases de datos guardadas en la lista utilizando la herramienta de consola del Visual SourceSafe.

PLAN DE TRABAJO

ETAPA	DURACIÓN
ETAPA INICIAL	**1 días**
Estimación de tiempo y costo	1 días
ANALISIS	**5 días**
Análisis Actual	**2 días**
Entrevistas	0.5 días
Diagnóstico	0.5 días
Revisión	0.5 día
Aprobación del diagnostico	0.5 día
Propuesta	**3 días**
Corregir lo que el diagnóstico dice	0.5 días
Elaborar propuesta	1 días
Revisión propuesta	1 día
Aprobación propuesta	0.5 día
DISEÑO	**1.5 días**
Interfaz	1 día
Clases	0.5 día
PROGRAMACION	**5.5 días**
Codificación	3 días
Depuración	0.5 día
Corrección	1 día
Ayuda	1 días
PRUEBAS	**1 días**
Corrección	1 días
TOTAL DE DÍAS	**14 días**

PROBLEMAS

Problema #1

No existe un horario fijo para realizar los respaldos, estos se realizan cuando la persona encargada posee un tiempo disponible.

Problema #2

No se tiene un control sobre las bases de datos que se deben respaldar, el encargado solo lo hace a memoria.

Problema #3

Si los respaldos necesitan más espacio de lo requerido no existe un mecanismo que busque espacio entre los discos para guardarlos provocando que los respaldos no se realicen por falta de espacio.

POSIBLES SOLUCIONES A LOS PROBLEMAS

Solución al problema #1

Crear un programa que realice los respaldos automáticos programándole el horario.

Solución al problema #2

Mantener en el programa la lista de las bases de datos a respaldar, con la posibilidad de incluir o borrar de la lista las bases de datos.

Solución al problema #3

Realizar un programa que tome cada base de datos que va a respaldar, verificar el espacio de la unidad de la ruta especificada para los respaldos, si no existe espacio buscarlo en las otras unidades.

ESPECIFICACIÓN DETALLADA DE LOS CASOS DE USO

#1. Mantenimiento de la lista de las bases de datos

Actores:

Administrador del sistema.

Propósito:

Ingresar, borrar bases de datos de la lista de bases de datos a respaldar.

Descripción:

El administrador del sistema ingresa o borra las bases de datos de la lista que se utiliza para generar los respaldos.

Tipo:

Primario.

Referencias cruzadas:

Curso normal de los eventos

ACTOR	SISTEMA
Asumiendo que el programa se encuentra abierto, el usuario hace clic en el botón añadir	El sistema muestra una venta para que el usuario navegue por las carpetas del disco duro.
El usuario selecciona la ruta de la base de datos del Visual SourceSafe	Verifica que no se encuentre incluida en la lista, la inserta y guarda la lista en el disco duro. El sistema lista las bases de datos en un grid y deja abierta una celda para que el usuario digite la clave y el usuario de la base de datos abierta.
El usuario digita la clave y el usuario de la base de datos abierta.	El sistema guarda la lista en el disco duro.

Una vez la lista hecha el usuario puede generar un archivo por lote que realice los respaldos. (ver Generar archivo por lote)	

Generar archivo por lote

Actor	Sistema
El usuario da clic en el botón Generar Script	**El sistema abre una dialogo para que el usuario seleccione la ruta de donde desea guardar archivo por lote.**
El usuario selecciona la ruta.	**El sistema genera el archivo y lo guarda en la ruta especificada.**

#2. Generar los respaldos

Actores:

El sistema, el usuario, el sistema operativo.

Propósito:

Con base a la lista guardada en el disco duro el sistema genera los respaldos.

Descripción:

El administrador del sistema ingresa programa una Tarea Programada en el sistema operativo para que el sistema se levante cada cierto tiempo y realice los respaldos

Tipo:

Primario.

Referencias cruzadas:

Curso normal de los eventos

ACTOR	SISTEMA
El usuario crea la tarea programada, para que el sistema se ejecuta solo cuando se desee.	
El sistema operativo ejecuta la tarea programada	El sistema levanta la lista de bases de datos guarda, verifica que existan, verifica el espacio disponible y genera los respaldos
Al final de la ejecución el usuario revisa y quema los respaldos.	

2.2 ALCANCE #2

En esta sección se describen los pasos para satisfacer el siguiente alcance: Generar los distintos instaladores de los productos SAF y e l instalador beta SYSDE Pensión y SYSDE Banca

- Asistir a una capacitación básica sobre la creación de inhaladores utilizando InstallShield X version 10.

- A base de la capacitación investigar sobre el lenguaje, sintaxis, configuración, conceptos en materia de creación de instaladores con InstallShield X version 10

- Estudiar los instaladores que se hayan generado anteriormente.

- Generar los instaladores, utilizando como base los instaladores creados anteriormente para productos SYSDE.

- Documentar el proceso de creación de los instaladores.

2.3 ALCANCE #3

En esta sección se describen los pasos para satisfacer el siguiente alcance: Actualizar el Marco de Gestión de Activos SYSDE SAF, acorde a los procedimientos documentados en la administración del repositorio y mediante un análisis GAP entre el Marco de Gestión de Activos SYSDE SAF y sistemas basados en Developer generar el borrador del Marco de Gestión de Activos para los sistemas Developer.

- Actualizar el Marco de Gestión de Activos SYSDE SAF, acorde a los procedimientos documentados en la administración del repositorio.

    - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.

    - Listar las diferencias del marco de gestión de activos con respecto a los documentos generados para la administración del repositorio.

    - Mediante esta lista actualizar el documento Marco de Gestión de Activos. Tomando cada aspecto diferente y describiéndolo.

- Mediante un análisis GAP entre el Marco de Gestión de Activos SYSDE SAF y sistemas basados en Developer generar el borrador del Marco de Gestión de Activos para los sistemas Developer.

    - Estudiar el marco de gestión de activos proporcionado vigente en la empresa.

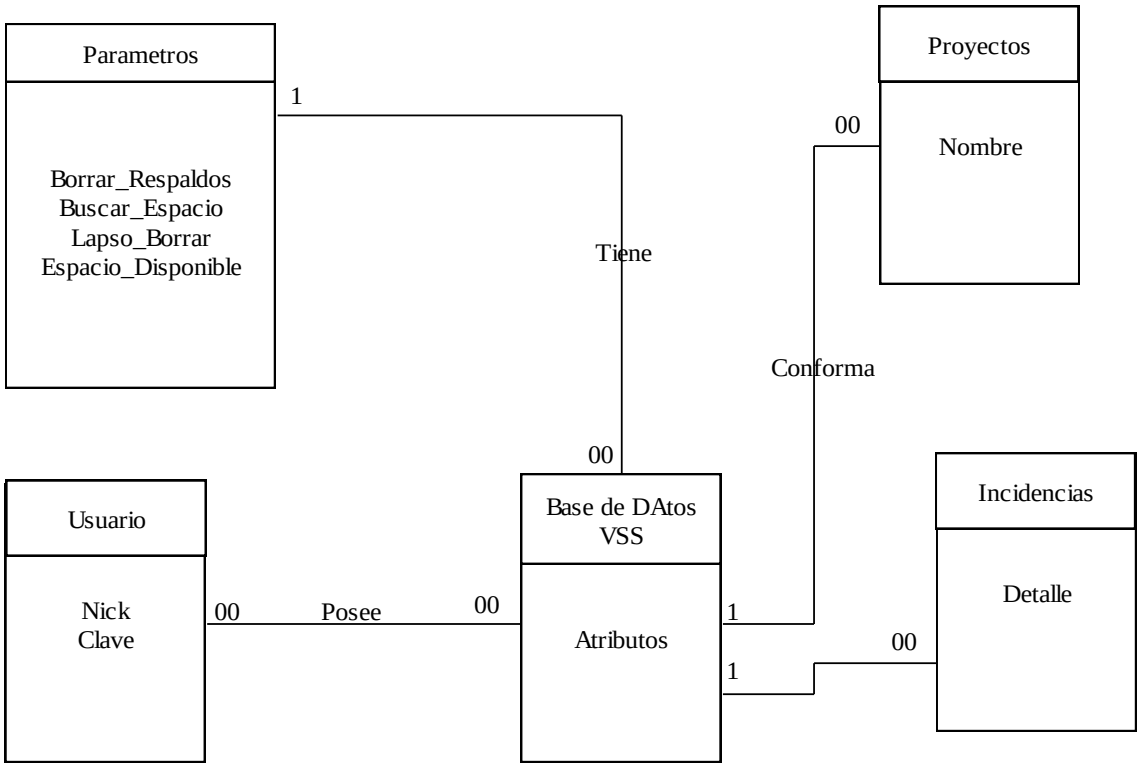
    - Listar las diferencias del marco de gestión de activos con respecto a los componentes de los sistemas basados en Developer.

    - Mediante esta lista determinar los componentes del Marco de Gestión de Activos que puedan ser funcionales para los sistemas desarrollados en Developer.

    - Desarrollar los componente faltantes para lo sistemas desarrollados en Developer, actualizar los heredados del Marco de Gestión de Activos.

3 DISEÑO DE LA HERRAMIENTA PARA REALIZAR LOS RESPALDOS AUTOMÁTICOS

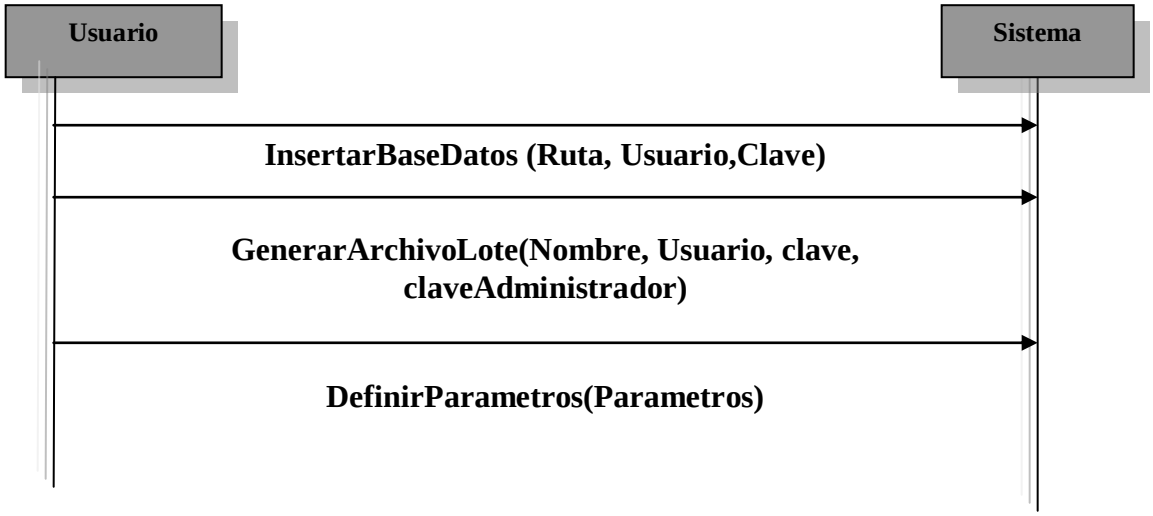
3.1 Modelo Conceptual



3.2 Diagramas de Secuencia

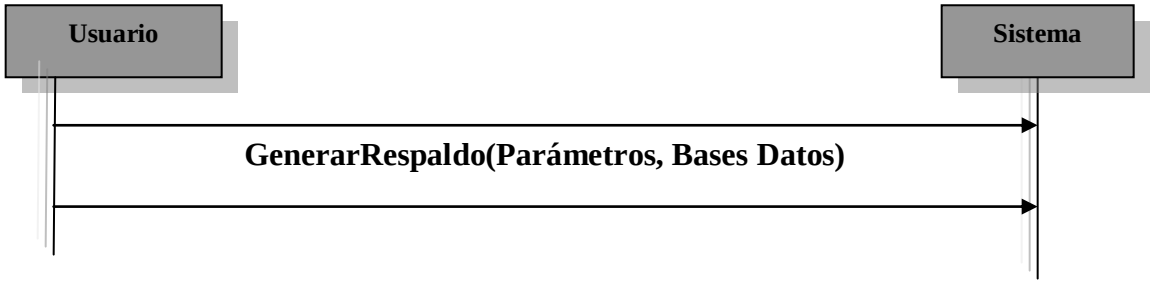
Caso de uso: Mantenimiento de la lista de la bases de datos

Diagrama de Secuencia



Caso de uso: Generar los respaldos

Diagrama de Secuencia



3.3 Modelo de Clases

Respaldo
Se encarga de la incidencias y lo procesos de respaldo

```
/// <summary>
/// guarda los eventos en el archivo
/// </summary>
/// <param name="evento">evento a guardar</param>
public void log(string evento)
{

                writer.WriteLine(evento);
                writer.Close();
}

/// <summary>
/// Genera las nuevas claves apartir de las actuales
/// </summary>
/// <param name="bd">nombre de la base de datos</param>
/// <param name="user">usario administrador de la base de datos</param>
/// <param name="clave">clave actuañ</param>
/// <returns>la clave nueva generada</returns>
public string NuevaClave(string bd, string user, string clave)
{
     return nueva;

}

/// <summary>
/// guarda las claves generadas en un archivo
/// </summary>
// <param name="fileName">nombre del archivo</param>
/// <param name="txtDirRespaldo"></param>
/// <param name="LB"></param>
public void generarClaves(string fileName,ListaBases LB)
{

     writer.WriteLine("Se ha finalizado la generación de claves");
     writer.Close();
}

/// <summary>
/// verifica que la clave tenga el formato correcto
/// para que la siguiente pueda ser generada
/// </summary>
/// <param name="clave"></param>
/// <returns></returns>
public bool parsear(string clave)
{
                return true;
}

/// <summary>/// Ejecuta los respaldos segun la lista de las bases de datos
/// </summary>
/// <param name="comando">Lo que se ejecuta</param>
/// <param name="argumento"></param>
/// <param name="varUser">usuario</param>
public  void Ejecutar(string comando , string argumento, string varUser)
{


}

/// <summary>/// Ejecuta los respaldos segun la lista de las bases de datos
/// </summary>
/// <param name="comando">Lo que se ejecuta</param>
/// <param name="argumento"></param>
/// <param name="varUser">usuario</param>
public void EjecutarListas( string txtDirRespaldo,ListaBases LB)
{

}
```

Archivos

```
    public bool borrarArchivos;  //para cargar del archivo se pueden borrar archivos de respaldo anteriores
    public bool otrasUnidades; //para cargar si se puede buscar espacio en otras unidades
    public long TiempoBorrar; //para cargar los dias, que detetmina que tan viejos tienen que ser los archivos para borrarlos
    public long EspacioDisco; //carga el tamaño en megas para determinar que unidad tiene espacio disponible
```

```
/// <summary>
/// guarda los eventos en el archivo
/// </summary>
/// <param name="evento">evento a guardar</param>
public void log(string evento)
{

}


        /// <summary>
        /// Genera las nuevas claves apartir de las actuales
        /// </summary>
        /// <param name="bd">nombre de la base de datos</param>
        /// <param name="user">usario administrador de la base de datos</param>
        /// <param name="clave">clave actuañ</param>
        /// <returns>la clave nueva generada</returns>
        public string NuevaClave(string bd, string user, string clave)
        {
;

        }

        /// <summary>
        /// guarda las claves generadas en un archivo
        /// </summary>
        /// <param name="fileName">nombre del archivo</param>
        /// <param name="txtDirRespaldo"></param>
        /// <param name="LB"></param>
        public void generarClaves(string fileName,ListaBases LB)
        {

        }

        /// <summary>
        /// verifica que la clave tenga el formato correcto
        /// para que la siguiente pueda ser generada
        /// </summary>
        /// <param name="clave"></param>
        /// <returns></returns>
        public bool parsear(string clave)
        {

        }

         /// <summary>
        /// carga un treeview con los items de la base de datos abierta
        /// </summary>
        /// <param name="lbItems">el listbox de donde se cargare el treeview</param>
        /// <param name="n">el treeview a cargar</param>
        /// <param name="i">la posicion del listbox</param>
        public void LlenarTV(System.Windows.Forms.CheckedListBox lbItems, TreeNode n, int i)
        {


        }


        public TreeNode BuscarNodo(string item, TreeNode n)
        {

        }


        /// <summary>
        /// Se utiliza para eejecutar el programa SS del SourceSafe, para listar (dir)
        /// los proyectos de una base de datos y cargarlos a un listbox
        /// </summary>
        /// <param name="lbItems">listbox donde se cargaran los proyecots listados</param>
        /// <param name="comando">programa a ejecutar (SS)</param>
        /// <param name="argumento">argumento del programa a ejecutar</param>
        /// <param name="varUser">valor de la variable de entonrno a modificar</param>
        /// <returns></returns>
        public  bool Ejecutar(System.Windows.Forms.CheckedListBox lbItems, string comando, string argumento, string varUser)
        {


        }


        /// <summary>
        /// genera el script con la lista de BD
        /// </summary>
        /// <param name="fileName"></param>
        /// <param name="txtDirRespaldo"></param>
        /// <param name="LB"></param>
        public void GenerarScriptDeListas(string fileName, string txtDirRespaldo,ListaBases LB)
        {


        }

        public int IndexArreglo(int[] arreglo ,int valor)
        {
            for (int i=0; i<=arreglo.Length; i++)
                if (arreglo[i]==valor)
                    return i;

            return 0;
        }
```

4 SISTEMATIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE

4.1 Naturaleza del Proyecto

Una de las necesidades más críticas para SYSDE consiste en la implementación de medidas estructurales para asegurar su capacidad de escalabilidad para la atención de proyectos en el contexto de un periodo de rápido crecimiento en sectores tales como fondos de pensión, banca e intermediarios financieros.

Los proyectos que SYSDE realiza en estos sectores incluyen primordialmente la implementación de sus productos existentes pero además como valor agregado se ofrece al cliente la oportunidad de adaptar nuestros productos a sus necesidades o requerimientos particulares.

Luego de estas implementaciones, pueden presentarse nuevas necesidades del cliente para aumentar el valor de los productos adquiridos lo cual crea una oportunidad para aumentar la funcionalidad de los productos.

Tanto la adaptación inicial de nuestros productos como las posteriores necesidades de extensión de funcionalidades expresadas por el cliente, desembocan en proyectos de desarrollo de software, cuyo éxito y rentabilidad deben ser asegurados por medio de la sistematización de los procesos para llevarlos a cabo.

SYSDE ha venido avanzando en la sistematización del Proceso de Desarrollo de Software para las situaciones comentadas, uno de los habilitadores claves de esta visión ha sido la puesta en marcha progresiva de una iniciativa cuyo objetivo es implantar un paradigma de fábrica de software basado en el concepto de los Software Product Lines (SPL) del Software Engineering Institute (SEI).

En proyectos anteriores se han desarrollado varios componentes tecnológicos que soportan la recolección de requerimientos utilizando la metodología RUP y ciertos procesos de la fábrica

de software de SYSDE. Este proyecto consiste en apoyar el desarrollo de los componentes tecnológicos que están pendientes para completar el proceso de Desarrollo de Software conjuntando la recolección de requerimientos y los procesos de fábrica ya automatizados, automatizando las partes del proceso faltantes, documentando las herramientas creadas en torno al proceso completo así como proponiendo e implementando un modelo automatizado para dar la propiedad de multi-lenguaje a los componentes tecnológicos.

4.2 Antecedentes

SYSDE considera que buena parte de su liderazgo y ventaja competitiva proviene de la instauración de procesos y el apoyo de los mismos a través de herramientas especializadas. El tema del proceso de desarrollo de software y la fábrica de software no es una excepción.
La sistematización del proceso de desarrollo de software y la gestión de los proyectos de este tipo involucra la introducción de herramientas especializadas para apoyar cada una de las disciplinas asociadas al proceso.

SYSDE le da una gran importancia al tema de las herramientas que apoyan sus procesos de fábrica. La razón de esto es que buena parte de la escalabilidad de la fábrica misma proviene de una selección cuidadosa de estas herramientas y su apropiada institucionalización.

Algunas de estas herramientas provienen de terceros que se han consolidados como estándares de facto a nivel de industria. Sin embargo, SYSDE ha encontrado necesario incursionar en el desarrollo de herramientas propias en ciertas áreas en donde las herramientas de mercado no están acordes con la naturaleza de su negocio o simplemente son inexistentes.

El punto de arranque del proceso de desarrollo de software e SYSDE es su proceso de gestión de requerimientos basado en casos de uso. En este contexto, SYSDE ya ha desarrollado una herramienta especializada que es utilizada por los consultores funcionales de campo para documentar los requerimientos con el cliente. También sirve como base para realizar el

análisis de brecha que permite determinar la amplitud de las adecuaciones que deben aplicarse al producto base para satisfacer las necesidades de un cliente particular.

Esta herramienta, conocida como *SYSDE Requisite Modeler*, ha sido concebida como un ambiente "stand-alone" que genera proyectos XML que son fácilmente intercambiables entre la fábrica y los proyectos *in situ* en las diversas geografías en donde opera la compañía. Muy ligado al *Requisite Modeler* se ha automatizado una parte del proceso de consolidación de esos modelos de casos de uso individuales basados en XML en una base de datos relacional centralizada que será el punto de partida para el resto de las herramientas del procesos de desarrollo de software.

El otro proceso que se ha automatizado en una versión base, es el de fábrica pero enfocado a la corrección de errores y gestión de requerimientos de corto alcance sobre los productos ya instalados.

4.3 Objetivos

Proponer e implementar un modelo lo más automático posible para proveer a las herramientas web desarrolladas de la propiedad multi-lenguaje utilizando las bondades de la plataforma .NET que incluye mecanismos para lograr este objetivo eficientemente.

Completar la herramienta de consolidación de modelos de casos de uso en formato XML que los incluye en un repositorio centralizado (una base de datos relacional) que sirve de base para el sistema de fábrica.

Apoyar el desarrollo de la integración de los modelos de requerimientos con el sistema de fábrica base ya desarrollado. Ampliar la funcionalidad del sistema de fábrica para que contemple los pasos pendientes de automatizar del proceso de desarrollo de software.

Documentar la herramienta de modelado de requerimientos desde un punto de vista metodológico utilizando los conceptos de casos de uso y RUP.

4.4 Alcances

Investigar e implementar un método proveer la propiedad multi-lenguaje a la Plataforma SYSDE, utilizando los archivos de recurso proveídos por la plataforma .NET.

Crear un manual de utilización de la herramienta de modelado de requerimientos desde un punto de vista metodológico utilizando los conceptos de casos de uso y RUP.

Apoyar el desarrollo de la integración de los modelos de requerimientos con el sistema de fábrica base ya desarrollado.

5 ESPECIFICACIÓN DEL PROYECTO SEGÚN ALCANCES PARA LA SISTEMATIZACIÓN DEL PROCESO DE DESARROLLO DE SOFTWARE

5.1 ALCANCE #1

En esta sección se describen los pasos para satisfacer el siguiente alcance: Investigar e implementar un método para proveer la propiedad multi-lenguaje a la Plataforma SYSDE, utilizando los archivos de recurso proveídos por la plataforma .NET.

5.1.1 Documentación y propuesta de método de multilenguaje.

- Estudiar como maneja e implementa .NET los archivos de recuros
- Investigar y documentar métodos para aplicar multilenguaje por medio de archivos de recurso.
- Probar los diferentes métodos y documentar las incidencias.

- Seleccionar un método en base a la documentación seleccionada, probarlo y documentarlo.
- Realizar una herramienta para la administración de archivos de recursos, aplicables a multilenguaje

5.1.2 Desarrollo de una herramienta para el manejo de archivos de recurso

OBJETIVO GENERAL

Realizar un programa que administre los archivos de recursos que proporciona .NET para aplicarlos a la funcionalidad de multilenguaje en las aplicaciones creadas en .NET

OBJETIVOS ESPECIFICOS

- Crear y mantener los archivo de recuso.

- Proveer opciones para una mantenimiento adecuado y acorde a la funcionalidad multilenguaje por aplicar.

- Asegurar el contenido no repetido y la nomenclatura de los archivos de recurso para aplicarlos como una herramienta en la programación en .NET, para dar la funcionalidad de multi lenguaje a cualquier aplicación que lo requiera.

DESCRIPCIÓN DEL PROYECTO

Realizar un programa que administre archivos de recursos utilizados, para la aplicación de la funcionalidad de multilenguaje para las aplicaciones creadas en .NET que los requieran.

Los archivos deben crearse nombrándolos con la nomenclatura que necesita .NET para aplicarlos a la funcionalidad multilenguaje. Esta nomenclatura requiere que el nombre de cada archivo posea un nombre inicial para identificarlos como grupo, seguido de la referencia cultural.

Una vez creado los archivos, el programa tendrá la opción de incluir ítems en cada uno de los archivos del grupo en conjunto. Así como realizar una búsqueda de ítems similares para descartar repetidos, pues el programa no deberá insertar ítems repetidos.

El programa tendra una opción para combinar dos archivos de recurso, uno de un grupo especifico y cualquier otro, con el propósito de pasar los ítems no repetidos de un archivo a otro.

ALCANCES

6. Crear grupos de archivos de recurso referenciado las culturas especificas al multileguaje en sus nombres.

7. Incluir y borrar ítems en cada una de los archivos de recurso del grupo.

8. Proveer una búsqueda avanzada de los ítems de cada archivo, con el fin de descartar repetidos

9. Crear nuevos archivos de recurso para un grupo especifico.

10. Combinar ítems de un archivo de recuro de un grupo con cualquier otro archivo de recurso.

PLAN DE TRABAJO

ETAPA	DURACIÓN
ETAPA INICIAL	**1 días**
Estimación de tiempo y costo	1 días
ANALISIS	**6.5 días**
Análisis Actual	**2.5 días**
Entrevistas	0.5 días
Diagnostico	1 días
Revisión	0.5 día
Aprobación del diagnóstico	0.5 día
Propuesta	**4 días**
Corregir lo que el diagnóstico dice	0.5 días
Elaborar propuesta	2 días
Revisión propuesta	1 día
Aprobación propuesta	0.5 día
DISEÑO	**3 días**
Interfaz	1 día
Clases	2 día
PROGRAMACION	**8 días**
Codificación	4 días
Depuración	1 día
Corrección	1 día
Ayuda	2 días
PRUEBAS	**1 días**
Corrección	1 días
TOTAL DE DÍAS	**19.5 días**

PROBLEMAS

Problema #1

Debido a que existen varios programadores en un proyecto, el manejo sincronizada y controlado de los archivos de recurso se vuelven muy tediosos, pues por cada programador existe un copia de archivos de recurso, y para mantener en orden los ítems de los archivos de recurso, se tendrían que digitar todos los ítems en cada archivo de recurso de cada programador.

Problema #2

El control de los ítems repetido no es parte del editor de archivos de recursos de .NET por lo que es fácil equivocarse, e insertar ítems con el mismo *Key* con diferente. Esta situación generaría un problema a la hora de generar los ensamblado de los archivos o provocaría la obtención de un ítems erróneo para una traducción

Problema #3

Los grupos de archivos para generar una traducción en multilenguaje deben tener los *key en los* ítems exactamente iguales, el key es el la llave para identificar la traducción. Esto provoca trabaja doble, triple o más dependiendo de la cantidad de archivos, pues hay que reescribir los *Rey* en cada archivo del grupo.

POSIBLES SOLUCIONES A LOS PROBLEMAS

Solución al problema #1

Para solucionar este problema se ha resulto realizar un programa que combine diferentes archivos de recurso funcionándolos dejando los ítems repetidos. De esta

forma, cualquier programador podrá trabajar de forma local, desentendiéndose de no escribir ítems que ya otro programador a escrito.

Solución al problema #2

Este problema se soluciona creando un programa que controle la inserción repetida de ítems, además de proveer un búsqueda de ítems similares para que el usuario pueda comparar entre ítems.

Solución al problema #3

Para solucionar este problema se ha resuelto crear un programa que maneje o administre los grupos de archivos de recurso, llevando el control de la inserciones en todos los archivos, tan solo escribiendo en uno.

SUPUESTOS

Supuesto #1

Se supone que los programadores mantendrán una copia local de los archivos de recurso, para el trabajo en su parte del proyecto.

Supuesto #2

Se supone que existe un persona que coordina los archivos de recurso para todo el proyecto, esta persona es la encargada de combinar todos los archivos de todos los programadores, para formar uno solo. Luego probar el proyecto globalmente.

ESPECIFICACIÓN DETALLADA DE LOS CASOS DE USO

#1. Mantenimiento de Ítems

Actores:

Administrador del sistema.

Propósito:

Ingresar borrar ítems de los archivos.

Descripción:

El administrador del sistema ingresa o borra los ítems de los archivos de recurso, si es necesario puede realizar una búsqueda de similares.

Tipo:

Primario.

Referencias cruzadas:

Curso normal de los eventos

ACTOR	SISTEMA
1. El usuario crea los archivos de recurso. (ver Mantenimiento de los archivos de recurso) o bien abre un grupo existente(ver sección, Abrir archivos de recursos)	El sistema carga todos los archivos y muestra solo un en un grid, permitiendo la opción de insertar ítems en el mismo gris.
2. El usuario elije el archivo al que desea insertarle un ítem	3. El sistema muestra ese archivo, mueve el cursor hasta el final de la grid.
	5. El sistema muestra tres formas diferentes de insertar un ítem, la primera es mediante el grid que contiene los ítems, la segunda mediante dos cajas de

	texto, una donde se especifica el *Key* y la otra donde se especifica el valor y por ultima un grid, donde se pueden especificar tanto el *Key* como el valor de un ítem.
	El sistema muestra una opción por medio de un *check* que especifica si el ítems a insertar se guardara solo en el archivo seleccionado o todos los del grupo.
El usuario elige la opción ya sea insertar en solo en el archivo seleccionado o en todos los del grupo.	
El usuario elije cual quiera de las opciones que desee para insertar un ítem, Para elegir la opción del grid el usuario hace clic en el botón “Insertar nuevo ítem” que se encuentra debajo del gris.	Si el usuario selecciona el la opción del grid el sistema crea una nueva fila en el grid para que el usuario inserte el ítem.
El usuario escribe los datos. En el caso de de las opciones que no sean las del grid que contiene los ítems, el usuario da clic en el botón “Insertar nuevo ítem”, el caso de la opción del grid el usuario ya sea presiona “enter”, o da clic en cualquier fila del gris.	El sistema verifica que ese ítems no se encuentre ya en el archivo de recurso seleccionad e inserta este en dicho archivo, y si la opción de insertar en grupo esta marcada el sistema guarda ese ítem en todos los archivos del grupo.
Si el usuario desea borrar un ítem, solo selecciona la fila en el grid, donde se encuentra el item	El sistema muestra un mensaje de confirmación
El usuario acepta la confirmación	El sistema borra el ítem seleccionado del archivo seleccionado y si la opción de insertar en grupo esta marcada el sistema borra ese ítem de todos los archivos del

	grupo.

#2. Mantenimiento de los archivos de recurso

Actores:

Administrador del sistema.

Propósito:

Descripción:

El administrador del sistema crea los archivos de recursos que sean necesarios agrupándolos y nombrándolos con la referencia cultural especificada.

Tipo:

Primario.

Referencias cruzadas:

Curso normal de los eventos

ACTOR	SISTEMA
1. El administrador selecciona la opción del menú ”Crear nuevos archivos”	El sistema despliega una ventana conteniendo un *checboxlist* con la lista de todas las culturas del sistema, y una caja de texto para que el usuario escriba el nombre del grupo
2. El usuario selecciona las culturas que desea, chequeándolas. Y escribe el nombre del grupo	3. El sistema crea por cada cultura seleccionada, un archivo, este archivo es nombrado por el sistema con el nombre digitado por el usuario más un punto y la referencia cultural

	El sistema carga todos los archivos, muestra la lista de estos en un *Treeview* y selecciona uno para mostrarlo en el grid.
Una vez las cultura cargadas el usuario puede: Abrir archivos de recursos (ver sección Abrir archivos de recursos), Crear nuevos archivos de recurso (ver Crear nuevo archivo de recurso referenciado una cultura), Combinar archivos de recurso (ver Combinar archivos de recurso)	

Abrir archivos de recuros

Actor	Sistema
1. El administrador del sistema accede a la opción "Abrir archivos de recursos"	**2. El sistema muestra una ventana don le permite al usuario navegar sobre las carpetas.**
3. El administrador elige la ruta de donde se encuentres los archivos	**El sistema muestra los archivos encontrados en la ruta seleccionada, agrupándolos en un *TreeView* por su nombre de grupo.**
4. El administrador selecciona en el *TreeView* el grupo que desea abrir	**5. El sistema carga los archivos del grupo seleccionado y**

Crear nuevo archivo de recurso referenciado una cultura

Actor	Sistema
1. El administrador del sistema accede a la opción "crear nueva cultura"	**2. El sistema despliega una ventana conteniendo un *checboxlist* con la lista de todas las culturas del sistema, y la lista de las culturas actuales. Además la opción de copiar los datos de una cultura existente a la nueva cultura por medio de un *checkbox***
3. El administrador selecciona la cultura que desea crear, y selecciona la cultura que desee que se copie en la nueva cultura, si la opción de copiar esta activa	**El sistema crea el nuevo archivo de la nueva cultura copias los datos si la opción de copiar esta activa. Carga el archivo en el grid de los ítems.**

Combinar archivos de recurso

Actor	Sistema
1. El administrador del sistema accede a la opción "Combinar archivos"	**2. El sistema muestra una ventana don le permite al usuario navegar sobre las carpetas.**
3. El administrador elige el archivo que se combinara con uno del grupo que se encuentra abierto.	**El sistema muestra copia los ítems no repetidos en el los archivos de recursos del grupo abierto. Si la opción de guardar indica que solo se guarde en el archivo seleccionado, el sistema solo copia en esto los ítems a combinar.**

5.2 ALCANCE #2

Crear un manual de utilización de la herramienta de modelado de requerimientos desde un punto de vista metodológico utilizando los conceptos de casos de uso y RUP.

- Estudiar el funcionamiento de la aplicación que utiliza SYSDE para recopilar los requerimientos de los proyectos y demás.

- Estudiar los conceptos de caso de uso y la metodología RUP utilizada por SYSDE.

- Crear el manual de RUP aplicado a la herramienta que SYSDE ha creado para el proceso de recopilación de información el *SYSDE Requisite Modeler.*

5.3 ALCANCE #3

Apoyar el desarrollo de la integración de los modelos de requerimientos con el sistema de fábrica base ya desarrollado.

Consiste en integrar SYSDE Requisite Modeler con la herramienta SYSDE Plataforma, que los requerimientos recopilados en SYSDE Requisite Modeler se conviertan automáticamente en actividades que los técnicos tengan que resolver. Para esto se sigue la especificación que SYSDE plantea.

6 IMPLEMENTACION DE LA PROPIEDAD MULTI-LENGUAJE A SYSDE FABRICA.

Para implementar el multilengueje, en ASP.NET Microsoft a creado y recomienda utilizar los archivos de recurso, estos llevan la extensión ".resx". Esto se implemento en el sistema de Fabrica SYSDE, desarrollado en ambiente Web.

Estos archivos contienen una estructura definida y se pueden crear en cualquier proyecto del Visual Studio.NET, sin embargo para la finalidad de implementar el multilenguaje solo necesitamos sus dos atributos, el **Key** y el **Value,** el primero se utiliza para identificar las frases en los diferentes lenguajes y el segundo es el valor de las frases. Para la primera opción los archivos deben contener datos, los cuales son las frases en los diferentes idiomas con su identificados único.

A continuación se describirán dos métodos posibles para implantar el multilenguaje con archivos de recursos, y la escogencia del mismo entre estos dos métodos.

6.1 Multilenguaje con archivos de recuro ensamblados

La primera opción es trabajar con los archivos de recuso ensamblados, antes de la ejecución y acceder estos ensamblados en tiempo de ejecución. Pero antes es necesario en el ambiente multilenguaje mantener un control estricto sobre los formularios (información grafica relacionada con el lenguaje), con el fin de mantener una relación adecuada entre los datos del formulario y los archivos de recursos usados para multilenguaje.

6.1.1 Crear y ensamblar los archivos de recursos

Este primer paso requiere una idea general de los formularios. Con esta idea general se crean los archivos de recurso. Como debemos ensamblar los archivos de recurso, los cuales poseen la extensión resx, es necesario convertirlos a la extensión resource.

Los pasos a seguir se describe a continuación.

- Creamos un proyecto nuevo, y en la raíz de este proyecto para cada uno de nuestros archivos de recursos creamos un directorio que haga referencia a la “Cultura” que queramos representar. Es esencial que los directorios mantengan en sus nombres las referencias culturales, nombrándolos con el código del lenguaje o cultura (ejemplo "es-CO", "en-US") para que el CLR pueda utilizarlos.

- Para todos los lenguajes a usar en la implementación es necesario crear un archivo de recurso. Este archivo de recurso lo guardamos en el directorio correspondiente al lenguaje. Otra cosa a tomar en cuenta es nombrar todos los archivos de recurso con el mismos nombre (ejemplo: MiRecurso.resx). Así en todos los directorios existirá un archivo recurso nombrado de la misma forma. Más adelante se describe la manera de crear estos archivos de recurso y las consideraciones a tomar en cuenta.

- Cuando se han terminado, de crear todos los archivos de recuro, lo que procede es ensamblarlos Pero antes es necesario convertir los .resx en .resources. Para ello utilizamos la herramienta que nos provee el SDK de .NET llamada resgen la cual se accede por medio del **Command Promt** de VisualStudio .NET.

  Para utilizar resgen, nos ubicamos en el directorio del lenguaje, y usamos la siguiente estructura

  Nombre de comando + archivo .resx + {archivo destino. + Referencia cultural + extensión .resources}

- Teniendo los .resources lo que queda es ensamblarlos. Ubicados en la misma carpeta procedemos a usar la herramienta al. La estructura para la ejecución de

al.exe es un poco más compleja que Resgen.exe. A continuación describiré los parámetros utilizados para realizar el ensamblado satélite:

/t:lib: este parámetro le indica al comando que lo que se obtendrá como output será una librería.

/**embed**: es el parámetro por el cual se le pasa al comando el archivo de recurso que será embebido en el ensamblado satélite.

/**culture**:"en-US" à este parámetro le indica al comando al que el **assembly** satélite que se esta creando es para operar sobre la referencia cultural que se define. Este parámetro es esencial para la realización de assemblies que sean utilizados para manejo de cultura, puesto que si no se incluye el **assembly** será considerado sin referencia cultural y por lo tanto el CLR no podrá utilizarlo para el proceso de traducción. El código de referencia cultural debe ir entre comillas. Ej.: /culture:"en-US".

/out: este parámetro indica la salida, o sea, con que nombre será identificado el **assembly** que será generado, producto de haber utilizado la herramienta al.exe.

- El paso siguiente es copiar las carpetas que hacen referencia cultural en la carpeta **bin** del proyecto miltilenguaje.

- Para un manejo más eficiente de la aplicación es necesario poseer un lenguaje o cultura por **default**, para ello el CLR necesita tener un ensamblado en la raíz de la carpeta **bin** del proyecto.

  En primera instancia elegimos cual lenguaje por **default** utilizar y copiamos el archivo de recurso que corresponda a este lenguaje en la carpeta **bin**. Hacemos los pasos 3 y 4, solo que quitamos la referencia cultural.

6.1.2 Implementar los archivos de recurso en la aplicación.

Una ves ensamblaos los archivos se pueden utilizar en la aplicación por lo que el siguiente paso, es cargar los datos con la información que nos proveerá el **assembly** satélite correspondiente a la referencia cultural solicitada por el cliente.

Para ello, inicialmente lo que uno tiene que asegurarse es que aunque el ensamblado no coincida con la configuración regional del cliente, el sistema cargue el ensamblado satélite por **default**, por lo tanto, para ello, utilizaremos los eventos del global.asax en el cual inicialmente cargaremos dicho ensamblado en una variable de aplicación de la siguiente forma:.

```
using System.Reflection;
using System.Resources;
using System.Threading;
using System.Globalization;
...
 protected void Application_Start(Object sender, EventArgs e)
{
    Assembly a = Assembly.Load("MiRecurso");
    Application["RSM"] = new ResourceManager("MiRecurso",a);
}
```

En primer lugar, es necesario declarar los cuatro espacios de nombre que se muestran en el cuadro anterior, en el global.asax.

- **System.Reflection** referenciado este espacio de nombre tendremos acceso a la clase **Assembly**, la cual nos permitirá cargar nuestro **asembly** satélite por **default**, para ello,

utilizaremos el método Load el cual espera que le pasemos una cadena representando el nombre del **assembly** a cargar.

- **System.Resources** este espacio de nombre nos permitirá utilizar la clase **ResourceManager** la cual es la encargada de leer la información almacenada en los archivos de recurso que posee el **assembly** satélite. Esta clase posee una sobrecarga en su constructor que recibe como parámetro, el nombre del recurso al que se quiere acceder (esto es porque como habíamos mencionada anteriormente un **assembly** puede contener mas de un archivo de recurso embebido), y el nombre de la variable de tipo Assembly que posee cargado el assembly satélite que utilizaremos.

- **System Threading** con este espacio de nombre lo que podremos hacer es obtener el valor de la cultura que se encuentra seteada en el proceso actual, o sea en el proceso que se encuentra corriendo nuestra aplicación. Esto lo utilizaremos mas adelante para poder cambiar de cultura la aplicación, respecto de la configuración regional que posea el cliente.

- **System.Globalization** aquí es donde se encuentra la clase encargada del manejo de cultural de las aplicaciones.

En la aplicación para realizar el cambio de lenguaje es necesario utilizar el siguiente código:

```
try
{
    Thread.CurrentThread.CurrentCulture=newCultureInfo(Request.UserLanguages[0],false);
    Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture
    (Thread.CurrentThread.CurrentCulture.Name);
}
catch(Exception)
{
    Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture("en-US");
}
finally
{
    Thread.CurrentThread.CurrentUICulture = Thread.CurrentThread.CurrentCulture;
}
```

La primera línea después del "**try"**.Debemos saber que la aplicación se encuentra corriendo en un sub-proceso el cual posee una cierta configuración inicial, esta configuración inicial es la que será utilizada por el sistema en caso de que no se le realice ningún cambio en medio. Realizaremos las acciones correspondientes para que el CLR obtenga el ensamblado satélite correspondiente a la cultura del cliente. Dentro del sub-proceso se encuentra la propiedad "**CurrentThread**" la cual hace referencia al proceso actual y dentro de ella, la propiedad "**CurrentCulture**", la cual hace referencia a la cultura actual, para que la cultura de la aplicación cambie, de acuerdo con la configuración regional del cliente es necesario que de esta ultima propiedad mencionada se cree un objeto de tipo "**CultureInfo**" el cual recibirá como parámetro la cultura actual del usuario que esta realizando el **Request** por medio de "**Request.UserLangages[0**]", además al constructor es necesario que se le pase un segundo

parámetro de tipo bolean el cual es para que el constructor tome la configuración de cultura por **default** que posee el CLR, puesto que de lo contrario, si el usuario le realizó algún cambio a su configuración regional, el sistema tomará este cambio y como consecuencia podría surgir un error de incompatibilidad de cultura.

Luego, la siguiente línea de código nos permitirá, mediante el método estático "**CreateSpecificCulture**", de la clase **CultureInfo**, cambiar la cultura del sistema, por la que fue obtenida de la configuración regional del cliente. Para realizarlo, al método **CreateSpecificCulture** es necesario pasarle por parámetro el nombre de la cultura actual. Dentro de este mecanismo es cuando el CLR se encarga de buscar en la carpeta correspondiente a la cultura solicitada. De no encontrar la carpeta con los recursos para dicha cultura el CLR cargará la cultura por **default**. Cabe aclarar que aunque el CLR no haya encontrado la carpeta con los recursos necesarios para la cultura solicitada, igualmente todo lo que refiere a código de moneda, fechas y calendario se adecuara a la cultura solicitada por el cliente.

Finalizado este punto, el código terminará su ejecución en la instrucción "**finally**" del bloque donde al proceso actual se le cargará en la propiedad "**CurrentUICulture**" la cultura actual. Esto se hace con el fin de que tanto el código de moneda como, los manejos de fechas, calendarios y ordenamientos, cumplan con lo establecido por dicha referencia cultural.

En caso de ocurrir una excepción por algún motivo, en el bloque de código "**catch**" se cargará la referencia cultura "en-US". Esto es una de muchas cosas que se podría llegar a hacer al momento de una excepción, por lo tanto, tengan en cuenta que, no es necesario seguir este punto en forma obligada

6.1.3 Acceso a los archivos de recurso

Una ves cargados los archivos de recurso y sabiendo como cambiar de lenguaje por medio de estos, lo que queda es acceder a los valores o sea las frases en el idioma escogido por el usuario, para ello utilizaremos el siguiente código:

```
rm  = (ResourceManager)Application["RSM"];

label1.text      = rm.GetString("NewUser_Name");
label2.text   = rm.GetString("NewUser_LastName");
labe3.text    = rm.GetString("NewUser_Address");
```

Cabe destacar que para utilizar este método se deben tener etiquetas, para mostrar todas las frases a la hora de cargar las paginas

En primer lugar encontraremos un campo denominado "**rm**" el cual será declarado de tipo **ResourceManager** con un modificador de acceso **private**, como ya es de imaginarse su **scope** dentro de la clase será de tipo global puesto que esta declarado fuera del método. Noten que al momento de inicializar el campo con un valor, dicho valor es casteado a un tipo "**ResourceManager**", esto es por que en el global.asax el objeto **ResourceManager** fue cargado en una variable de aplicación la cual guarda al objeto de tipo "**Object**", por lo tanto, para que lo podamos utilizar, es necesario convertirlo a su tipo original. Es necesario declarar el espacio de nombre **System**.**Resources**, para poder trabajar con la clase **ResourceManager**.

Una vez inicializado el campo **rm** ya estamos en condición de cargar los valores del archivo de recurso a los campos. Para ello se utilizara el método **GetString** del objeto **rm**, el cual

recibe como parámetro el nombre del campo que quiero invocar del archivo de recursos que nos provee el **assembly** satélite.
Otra alternativa para acceder a los archivos de recuso se refiere a los casos en que no se quieran usar controles para desplegar texto en la pagina. Para ello se debe utilizar la parte del lenguaje HTML

Lo que debemos hacer inicialmente, es crear un campo de tipo **string** con un modificador de acceso **private** para cada uno de los datos que serán cargados desde el **assembly** satélite.

```
private string _name;
private string _lastName;
private string _address;
```

Una vez realizado los campos, es necesario crear una propiedad para cada campo, la cual será de solo lectura, retornará un **string** y su modificador de acceso será de tipo **protected**. Estas propiedades devolverán el valor que será cargado en los campos antes realizados.

```
protected string Name{get{return _name;}}
protected string LastName{get{return _lastName;}}
protected string Address{get{return _address;}}
```

Luego de haber definido las propiedades, es necesario volver al formulario de la aplicación, pero en lugar de ver el formulario desde su diseño, pasaremos a ver directamente el código HTML.

Ahora bien, para poder mostrar los valores que fueron cargados en los campos privados del **Code-Behind**, es necesario realizar un "**Bind**" de los datos, y para ello utilizaremos las propiedades antes definidas, la cuales serán invocadas desde el formulario HTML de la siguiente forma:

```
<TABLE>
        <TR>
                <TD><%#Name%>  :</TD>
...
```

De esta forma, al cargar con datos los campos privados estos nos darán la posibilidad de transferir esa información a la interfase de usuario para luego ser mostrada al cliente.

```
rm  = (ResourceManager)Application["RSM"];
_name       = rm.GetString("NewUser_Name");
_lastName   = rm.GetString("NewUser_LastName");
_address    = rm.GetString("NewUser_Address");
...
DataBind();
```

6.2 Pasos para implementar archivos de recursos en multi-lenguaje en ensamblando en tiempo de ejecución.

Este método difiere del anterior en el sentido de que los archivos de recurso de ensamblan en tiempo de ejecución, el resto de consideraciones en cuanto a la introducción de datos y referencias culturas no difieren tanto. A continuación se describirá como crear los archivos y accederlos dentro de la aplicación

6.2.1 Creación de los archivos de recursos

Los archivos de deben crear en el proyecto. Debemos recordar que debe existir un archivo para cada lenguaje o cultura que se utilizara en la aplicación, para ello:

- Estos archivos deben seguir una estructura especifica. Que llamaremos grupo.
  Resource + {codigo lenguaje }+.resx
  Ejemplos
  Resource.ca.resx Para catalan
  Resource.es.resx Para español
  Resource.en.resx Para ingles
- Una vez creados se almacena en una carpeta cualquiera dentro de la solución

6.2.2 Implementación de los archivos de recurso en la aplicación

A diferencia, del método anterior, este no necesita inicializar ninguna variable en el "global.aspx", ni requiere una ensamblado previo. Solo se necesita contener los archivos en una carpeta especifica en el proyecto del Visual Studio.NET, y proceder con lo siguiente:

- El primer paso es cambiar la cultura, del hilo actual en que esta corriendo la aplicación: Mediante **CodigoIdioma** se establece el idioma a usar por ejemplo ("kha" catalán), la siguiente línea de código permite especificar la cultura que utilizara la aplicación

```
'Establece el thread de cultura para formatos, comparaciones...
Thread.CurrentThread.CurrentCulture =
cinfo.CreateSpecificCulture(CodigoIdioma)
'Establece el valor de cultura para los satellite assemblies
Thread.CurrentThread.CurrentUICulture = cinfo.CurrentCulture
```

- El siguiente paso es cargar el ensamblado: Una ves especificada la cultura se utiliza una variable tipo **ResourceManager** declarada previamente para acceder los archivos de recurso

```
'Cargamos el archivo de resources
rm = New ResourceManager("AppMultiidioma.Resource",
System.Reflection.Assembly.GetExecutingAssembly)
rm.IgnoreCase = True
```

- Finalmente accedemos a los datos mediante el método **GetString**() del **ResourceManager**

```
Me.lblBienvenida.Text = rm.GetString("lblBienvenida")
Me.lblUsuario.Text = rm.GetString("lblUsuario")
Me.HyperLinkCastellano.Text = rm.GetString("lblCastellano")
Me.HyperLinkIngles.Text = rm.GetString("lblIngles")
Me.HyperLinkCatalan.Text = rm.GetString("lblCatalan")
```

6.3 Método escogido.

Tras conocer los dos métodos posibles, sus consideraciones y su forma de implementación, paso siguiente es seleccionar uno de ellos. En esta sección se describirá cual método se eligió y por que motivos se eligió.

6.3.1 Antecedentes

Es importante antes de escoger un método para proveer la propiedad multilenguaje a un proyecto ya avanzado como los SYSDE Fabrica, valorar la aspectos como si algunos de los aspectos utilizados en los métodos ya se encuentran implementados.

En el sistema, ya se encontraba implementado la utilización de archivos de recurso para el manejo de errores. Para este fin el sistema de fabrica mantenía un archivo de recurso donde se especificaban los errores por medio de una llave. Este archivo se ensamblaba en tiempo de ejecución, y se acedía a el como lo especifica el segundo método mencionado anteriormente.

Sin embargo, este archivo para el manejo de errores, no se relacionaba en ningún momento con la propiedad multilenguaje, solo se utilizaba para mantener los mensajes de error y accederlos por medio de una llave. No existían archivos que llevaran en sus nombres las referencias culturales, para que el ensamblado tome efecto cuando se cambiaba la cultura del hilo actual, que tampoco se cambiaba.

6.3.2 Las razones de la escogencia

Se decidió escoger el segundo método, debido a:

- Los antecedentes del proyecto que ya implantaba archivos de recurso por ese método.

- Facilidad, pues el primer método requiere un trabajo previo antes de correr la aplicación, y una puesta en producción tomaría mayor tiempo.

- El acceso de los datos no es recomendable desde la parte de HTML, debido a que las paginas de la aplicación están sobrecargadas de información, y seria muy tedioso mantener esta relación de la parte HTML con el **Code-Behind**. Por lo que se descarta la alternativa mencionada en el primer método.

6.3.3 Problemas

Una vez escogido el método surgió la pregunta de cual debería ser la llave en los archivos de recurso para acceder a las diferentes frases en los diferentes lenguajes, recordemos que la misma frase en diferente idioma debe contener una llave única.

El segundo problema consistía en como traducir tanto texto de las paginas. Si se había descartado el primer método, no se podía utilizar las sentencias HTML para poner el texto en las paginas. Por otro lado traducir etiquetas como lo muestra el segundo método resultaría muy complicado y laborioso

La cantidad de objetos utilizados en las paginas complicarían aun más la traducción si se hiciere como lo muestra el segundo método, en las etiquetas.

Una idea es recorrer todos los objetos de la pagina mediante un método, para que este pueda acceder el texto de cada objeto y cambiarlo al lenguaje especifico. Pero esto lleva a otro problema, como identificar cada texto, en cada objeto, pues los archivos de recurso se acceden por medio de una llave.

6.3.4 Implementación

Para resolver los problemas planteados anteriormente se ideo establecer la llave de los archivos de curso como la frase en español, es decir, en cada archivo, la llave contendría la frese en español y en el valor su correspondiente en el otro idioma. Esto evita la utilización de códigos, que auque eficientes resultan complicados a la hora del acceso.

Esta forma de establecer la llave elimina el problema de identificar los objetos para recorrerlos y cambiarles el texto, pues es solo tomar su valor en español. Sin embargo requiere que todo el texto de las paginas este contenido en objetos.

A continuación se describen los pasos para implementar el multilenguaje:

- Crear una variable de sección para mantener el lenguaje seleccionado por el usuario, pues la cultura debe cambiarse, cada ves que de produzca una acción en la pagina, o sea en el hilo actual de la aplicación.

- Crear los archivos de recurso, tal como lo especifica el segundo método. Se crearon tres grupos de archivo, los que manejan los errores (especifican los errores para mostrarlos al usuario), los que manejan los mensajes (contiene los mensajes de la aplicación), los que manejan las etiquetas (contienen todo el texto de los objetos que se cambia dinámicamente.

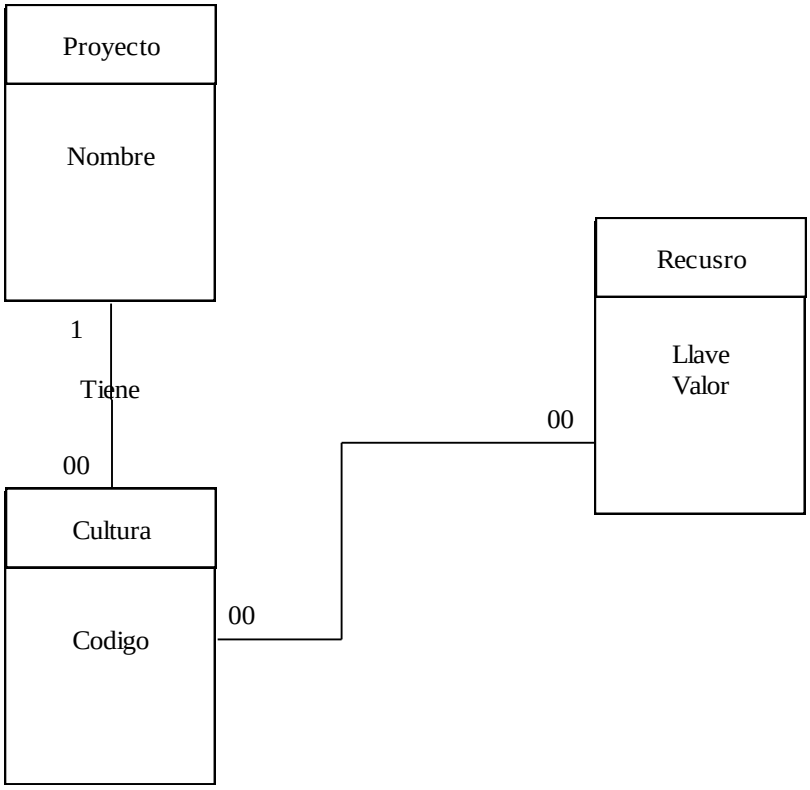
- Se creo un método para recorreré lo objetos de la pagina, que cambia de cultura al hilo, accede a los archivos y cambie el texto de los objetos. Este método recibe el objeto **Page** de la página, junto con la variable de sesión que contiene la cultura especificada por el usuario, esto para poder cambiar la cultura.

- Un método para obtener valores sobre una llave específica, para cada grupo de archivos de recurso.

- Cambiar todo el texto que se asigno por medio de sentencias de HTML a objetos, para que puedan ser cambiados por el método que recorre a estos.

7 DISEÑO DE LA HERRAMIENTA PARA REALIZAR LOS RESPALDOS AUTOMÁTICOS

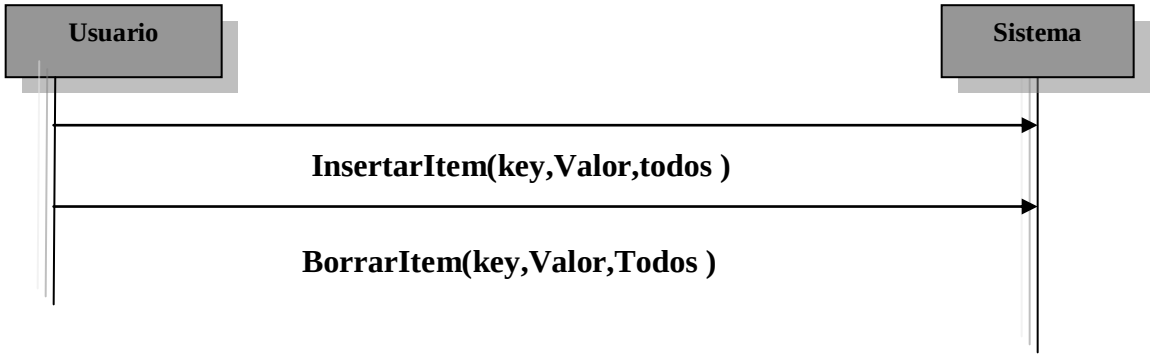
7.1 Modelo Conceptual



7.2 Diagramas de Secuencia

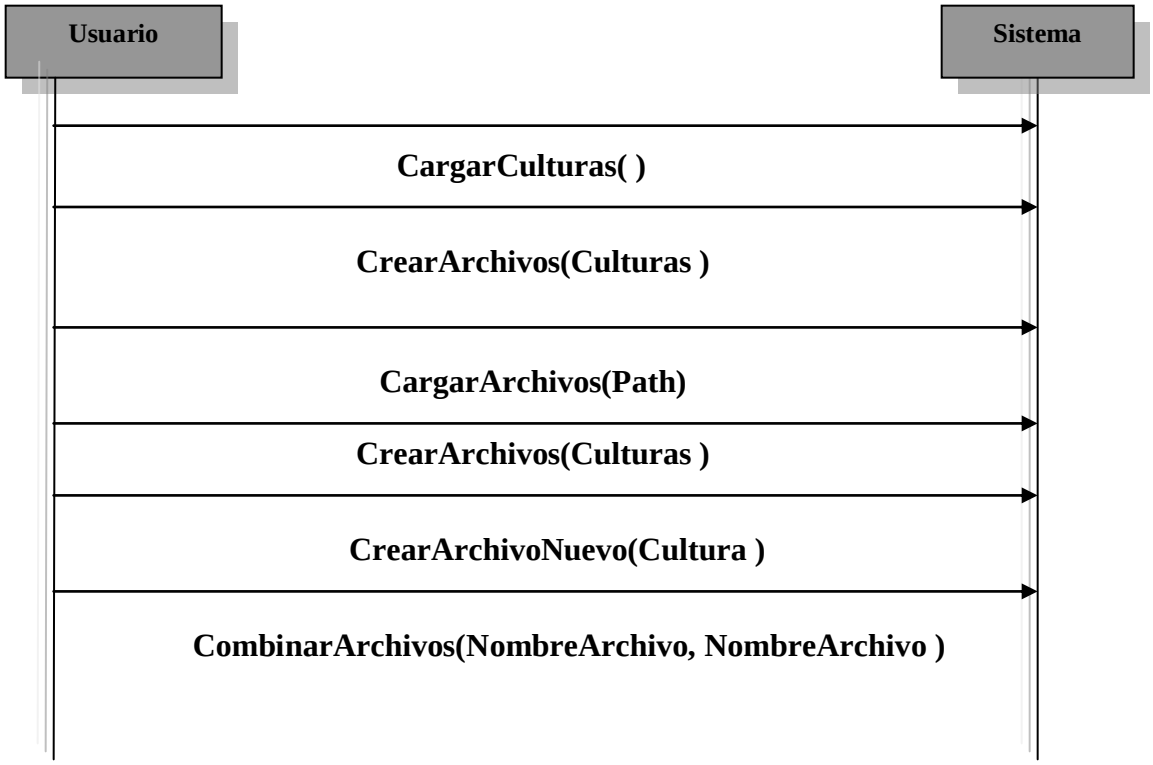
Caso de uso: Mantenimiento de Ítems

Diagrama de Secuencia



Caso de uso: Mantenimiento de los archivos de recurso

Diagrama de Secuencia



7.3 Modelo de Clases

Archivos

```
        /// <summary>
        /// caraca las culturas en el ListBox
        /// </summary>
        /// <param name="lbCulturas">List para cargar las culturas</param>
        public void ListarCulturas(ListBox lbCulturas )
        {

        }

        ///*************************************************************************

        /// <summary>
        /// Crea un tabla con los datos de un archivo de recurso
        /// </summary>
        /// <param name="Tabla">DataSet donde se añadira la tabla</param>
        /// <param name="Archivo">Archivo donde se toman los datos</param>
        public void CargarArchivosTabla(DataSet Tabla, string Archivo )
        {


        }
        ///*************************************************************************


        /// <summary>
        ///Lista todos los archivos de recurso que existan en un direcorio
        ///especifico, esto es un paso para agruparlos en un TreeView
        /// </summary>
        /// <param name="lbList">donde se lista</param>
        /// <param name="grupo">nombre del grupo de archivos, para buscarlos</param>
        /// <param name="directorio">El directorio donde se deben buscar</param>
        /// <returns>false si hay un fallo</returns>
        public bool ListarArchivos(ListBox lbList,string grupo, string directorio )
        {

        }
        ///*************************************************************************


        /// <summary>
        ///agrupo los archivos de recurso segun su nombre
        ///para que el usurio los escoja
        /// </summary>
        /// <param name="lbList"> el listbox donde se toman</param>
        /// <param name="tvList">el TreeView donde se agrupan</param>
        public void AgruparArchivos(ListBox lbList,TreeView tvList  )
        {
            //el lbList esta ordenado,se rrecorre, se extrae el nombre sin la cultura

        }

        ///*************************************************************************


        /// <summary>
        ///carga los archivos en las tablas, esto lo have recorriendo el TreeView
        ///y llamando a VerArchivos
        /// </summary>
        /// <param name="n">nodo que contiene el nombre de los archivos a cargar</param>
        /// <param name="Tabla">donde se cargaran los archivos</param>
        /// <param name="lbListC">Para saber que culturas se estan cargando</param>
        /// <param name="tvLenguaje">Donde se carga los nombres de las culturas que      son iguales a los nombres de las tablas del
DataSet</param>
        /// <param name="dgv">Se cargan en el los nombres de las culturas para especificar
        ///                  que debe incluir el usuario, este es el grid que sirve para incluir</param>
        /// <param name="directorio">Directorio donde estan los archivos</param>
        public void CargarArchivos(TreeNode n, DataSet Tabla ,CheckedListBox lbListC, TreeView tvLenguaje, UltraGrid dgv, string
directorio)
        {


        }



        //*****************************************************************
        /// <summary>
        /// Carga las culturas al tree view y cheque las culturas en el CheckedLsitBox
        /// </summary>
        /// <param name="namefile">nombre del archivo</param>
        /// <param name="lbList">lista de las culturas</param>
        /// <param name="tvLenguaje">nodo donde se añadira la cultura</param>
        /// <param name="dgv">grid de que sirve paara incluir, se le añade la cultura</param>

        public bool ObtenerIdioma(string namefile, CheckedListBox lbList, TreeNode tvLenguaje,UltraGrid dgv)
        {
            string [] idioma=namefile.Split('.'); //extrae la cultura del nombre del archivo

        }

        ///*************************************************************************

        /// <summary>
        ///compara las cadenas
        /// </summary>
        /// <param name="Key">lo que se va a buscar</param>
        /// <param name="KeyTablas"> en donde se busca</param>
        /// <returns>true si son similares</returns>
        public bool cadenaSimilar(string Key,string KeyTablas)
        {


        }

        ///*************************************************************************

        /// <summary>
        /// recorre la tabla para buscar Key´s similares, llamando a cadena similar
        /// y añade los similares a un listbox
        /// </summary>
        /// <param name="key">valor a buscar</param>
        /// <param name="tabla">tabla donde se buscaran</param>
        /// <param name="dt">donde esta la tabla</param>
        /// <param name="lb">donde se cargan los similares</param>
        public void buscarSimilares(string key , string tabla, DataSet dt, ListBox lb, bool buscarvalor)
        {

        }
```

```
///*************************************************************************************

        /// <summmary>
        ///
        /// </summmary>
        /// <param name="'Key''>DataSet Donde estab las tablas</param>
        /// <param name="'Value''>Valor nuebo</param>
        /// <param name="'tabla''>Tabla actual</param>
        /// <param name="'g''>opcion de agrupamiento</param>
        /// <returns></returns>
        public bool NuevoKey(DataSet dt, string Key, string Value, string tabla,bool g)
        {

        }
        ///*************************************************************************************

        /// <summmary>
        /// inserta un nuevo item a las tablas desde dos TextBox
        /// si la opcion agrupar es false solo inserta en la tabla actual
        /// </summmary>
        /// <param name="'dt''>DataSet donde estan las tablas</param>
        /// <param name="'tabla''>tabla actual</param>
        /// <param name="'Key''>key a introducir</param>
        /// <param name="'Value''>Valor a introducir</param>
        /// <param name="'g''>opcion de agrupamiento</param>
        /// <returns>retorna false si el key esta en la tabla actual</returns>
        public bool NuevoKeyGrid(DataSet dt, string tabla, string Key,string Value,bool g)
        {

        }
        ///*************************************************************************************

        /// <summmary>
        ///Rrecorre las tablas y las guarda en los archivos del gtupo especifico
        /// </summmary>
        /// <param name="'dt''>DataSet Donde estan cargados los datos</param>
        /// <param name="'grupo''>nombre del grupo al que pertenencen los archivos</param>
        /// <param name="'directorio''>Directorio donde se guardaran</param>
        public void SalvarArchivos(DataSet dt, string grupo, string directorio)
        {

            }
        }
        ///*************************************************************************************

        /// <summmary>
        /// Busca si hay un key repetido en la tabla
        /// </summmary>
        /// <param name="'dt''>Dataset con las tablas</param>
        /// <param name="'key''>Key a buscar</param>
        /// <param name="'tabla''>tabla actual</param>
        /// <returns>false= si no esta el key </returns>
        public bool RepetidosEnTable(DataSet dt,string key, string tabla)
        {

        }
        public bool RepetidosEnTableActualizar(DataSet dt,string key, string tabla)
        {

        }
        ///*************************************************************************************

        /// <summmary>
        /// recorre las tablas y actuliza la fila que tebga el
        /// key igual excepto la tabla actual, la opcion es agruopar
        /// no have nada
        /// </summmary>
        /// <param name="'dt''>DataSet con las tablas</param>
        /// <param name="'tabla''>Tabla actual</param>
        /// <param name="'Key''>Key nueva</param>
        /// <param name="'Value''>Value nuevo</param>
        /// <param name="'KeyUp''>Key viejo</param>
        /// <param name="'g''>opcion de agrupar</param>
        /// <returns>retorna false si en la tabla actual ya esta el Key </returns>
        public bool ActualizarKeyGrid(DataSet dt, string tabla, string Key, string Value, string KeyUp, bool g)
        {

            return true;
        }


        ///*************************************************************************************
        /// <summmary>
        /// Borra filas de las tablas del dataset, con el Key como id, si
        /// la opcion agrupar es false no se have nada
        /// </summmary>
        /// <param name="'dt''>DataSet con las tablas</param>
        /// <param name="'tabla''>_Nombre de la tabla actual</param>
        /// <param name="'key''>Key a borrar</param>
        /// <param name="'g''>opción agrupar</param>
        public void EkerrarFilas(DataSet dt, string tabla, string key, bool g)
        {
            if (g) //si la opcion agrupar esta activa
                    //recorre las tablas si encuentra el key en la tabla borra la fila
                for (int j=0 j<dt.Tables.Count;j++)
                    if (string.Compare(tabla,dt.Tables[j].TableName)!=0)
                        for (int i=0 i<dt.Tables[j].Rows.Count; i++)
                            if (string.Compare(dt.Tables[j].Rows[i].ItemArray[0].ToString(),key)==0)
                            {
                                dt.Tables[j].Rows[i].Delete();
                                dt.Tables[j].AcceptChanges();
                            }
        }
        ///*************************************************************************************

        /// <summmary>
        /// Busca archivos iguales a el grupo seleccionado
        /// esto para no crearlos
        /// </summmary>
        /// <param name="'path''> path del directorio+el grupo</param>
        /// <returns>true si hay archivos</returns>
        public bool BuscarArchivosResx(string path)
        {

        }
        ///*************************************************************************************
        ///Crea los archivos que el usuario seleccionó en el CheCheckedListBox
        ///donde estan las culturas
        public bool CrearArchivosNuevos(string nombre, CheckedListBox lbList)
        {

        }
        ///*************************************************************************************

        /// <summmary>
        /// Busca para ver si existe una cultura en el treeview
        /// </summmary>
        /// <param name="'tv''>treeview</param>
        /// <param name="'cultura''>Nombre de la cultura</param>
        /// <returns>retorna true si encuentra una cultura igual al parametreo cultura</returns>
        public bool ExisteCultura1V(TreeView tv, string cultura)
        {

        }

        ///*************************************************************************************

        /// <summmary>
        /// Crea una tabla y la añade al dataset
        /// </summmary>
        /// <param name="'Tabla''>DataSet donde se añadira la tabla</param>
        /// <param name="'Archivo''>Archivo donde se toman los datos</param>
        public void CrearTabla(DataSet Tabla, string nombreTabla )
        {

        }
        ///*************************************************************************************

        ///*************************************************************************************

        /// <summmary>
        /// Crea una tabla y la añade al dataset
        /// </summmary>
        /// <param name="'Tabla''>DataSet donde se añadira la tabla</param>
        /// <param name="'Archivo''>Archivo donde se toman los datos</param>
        public void CrearTablaCopia(DataSet Tabla, string nombreTabla , string TablaOrigen)
        {

        }
        ///*************************************************************************************

        /// <summmary>
        /// Busca si hay un key en el grid y devuelve el indece en la tabla
        /// </summmary>
        /// <param name="'dg''>Grid con las tablas</param>
        /// <param name="'key''>Key a buscar</param>

        /// <returns>retorna -1 si no esta el key</returns>

        public int activarcelda(UltraGrid dg,string key)
        {

        }

        /// <summmary>
        ///Rrecorre una tabla y las guarda en los archivos del gtupo especifico
        /// </summmary>
        /// <param name="'dt''>DataSet Donde estan cargados los datos</param>
        /// <param name="'grupo''>nombre del grupo al que pertenencen los archivos</param>
        /// <param name="'directorio''>Directorio donde se guardaran</param>
        /// /// <param name="'tabla''>tabla en la cual estab los datos</param>
        public void SalvarArchivos(DataSet dt, string grupo, string directorio, string tabla)
        {

        }
        public void CombinarArchivos(UltraGrid dgTables, string Archivo )
        {

        }
        ///*************************************************************************************
        public bool BorrarArchivo( string grupo, string directorio, string cultura, DataSet ds)
        {

        }
```

8 CONCLUSIONES

Comunicación, facilita el flujo del trabajo: Lo principal de trabajar en equipo es mantener una comunicación eficiente, que permita el buen flujo de información con el fin de no dejar ningún detalle y evitar malos entendidos.

Documentación, elimina dependencia a recursos: Si no se documentan los proyectos, y cada recurso se especializa en un área, la empresa se vuelve dependiente de estos recursos. Si faltase un recurso, el reemplazo le llevaría mucho más tiempo tomar el trabajo del recurso anterior.

Iniciativa, permite avanzar más rápidamente: La iniciativa es parte de una persona emprendedora, dispuesta y trabajadora, la cual puede llegar a objetivos importantes. Por otro lado la iniciativa evita el constante mandato de los superiores y agiliza el trabajo.

Seguir los planes de trabajo resulta difícil: No en todos los proyectos las cosa resultan como se planearon, lo más difícil de un proyecto es asignar horas recurso. Cualquier imprevisto, problema, se podría reflejar un atraso en los planes de trabajo. Que puede resultar difícil retomar.

Es importante poner atención a los detalles, estos pueden resultar importantes para tomar una decisión. También permiten a la persona ser más profesional y realizar las tareas de la mejor manera posible.

Es importante para los estudiantes, mejorar la redacción y ortografía, esto les permitirá mostrarse como verdaderos profesionales y contribuir a un trabajo de calidad.

Cada proyecto que se desarrolle debe pensarse en función de un sistema de calidad, por pequeño que fuese. Si la empresa dar un servicio de calidad, debe documentar todos los procesos e incidencias, así como diseñar los proyectos pensando en el futuro, en el sentido de un sistema de calidad.